# Artistic Style Transfer

Elias Wang[1], Nicholas Tan[1]

***Abstract*—We have shown that it is possible to achieve artistic style transfer within a purely image processing paradigm. This is in contrast to previous work that utilized deep neural networks to learn the difference between "style" and "content" in a painting. We leverage the work by Kwatra et. al. on texture synthesis to accomplish "style synthesis" from our given style images, building off the work of Elad and Milanfar. We have also introduced a novel "style fusion" concept that guides the algorithm to follow broader structures of style at a higher level while giving it the freedom to make its own artistic decisions at a smaller scale. Our results are comparable to the neural network approach, while improving speed and maintaining robustness to different styles and contents.**

***Index Terms*—Style Transfer, Image Processing, Texture Synthesis, Nearest Neighbor, Principal Component Analysis, Segmentation Mask**

## I. Introduction

THE recent success of Convolutional Neural Networks (CNNs) by Gatys et al. [4] in the Style-Transfer task has generated renewed interest in the topic.

Fig. 1: Gatys shows that CNN's can distinguish between style and content

Furthermore, with the advent of new platforms such as Snapchat and Instagram that create accessible avenues for art-technology to the public, the problem has become all the more relevant to the scientific community. With the motivation of reducing computational learning overhead as well as run time, there has been subsequent work to capture the flavor of the machine-learning process and differentiation between style and content in terms of pure image processing. Our effort has been in trying to improve the artistic quality of the style transfer within this space.

[1]Department of Electrical Engineering, Stanford University, Stanford, CA

## II. Related Work

One approach to style transfer is building off the related area of texture synthesis. Texture synthesis aims to generate a realistic texture image from a sample texture. In terms of the style transfer task, we can think of the style as the texture we are trying to mimic.

One successful texture synthesis algorithm was proposed by Kwatra et al. [2], which is based on minimizing an energy function. This method of global optimization is in contrast to other techniques that use region-growing.

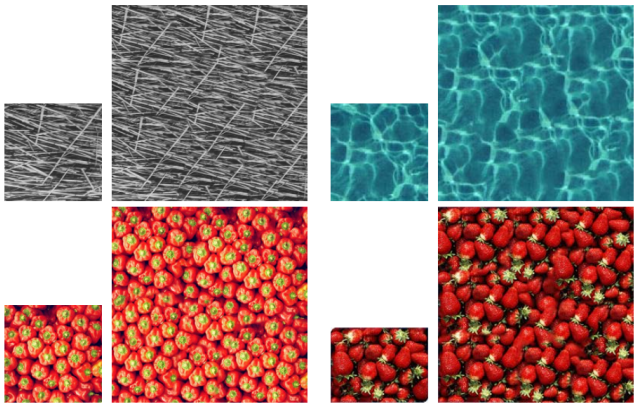

Fig. 2: Kwatra shows successful generation of various textures

Elad and Milanfar [1] adapted the ideas of Kwatra et al. to style transfer. By first deriving an analogous objective function for the energy minimization, then performing the optimization in stages, patches from a style image can be robustly and successfully mapped to regions in the content image, providing pleasing artistic results.

An alternative approach of achieving style transfer is through image component analysis, as demonstrated by Zhang et al. [5]. This approach decomposes the content and style images into the draft, paint, and edge components, then the paint and edge components of the content image are modified to match those of the style image, and then finally the final image is reconstructed from the new content image components.

## III. Methods

Our algorithm derives motivation from the recent work of Elad and Milanfar [1], which derives several stages of

optimization from an energy minimization point of view. We adapt these stages, while providing our own implementation.

### A. Overview

The core of the algorithm consists of six steps: style fusion, patch matching, style synthesis, content fusion, color transfer, and denoising. We apply this process for a series of decreasing scales and patch sizes, allowing the algorithm to make broad style transfers then progressively tune them. Each of these will be described in more detail in the following subsections.

### B. Terminology

Throughout this paper, we will refer to the series of energy functionals from Elad's work [1] in optimizing the texture synthesis algorithm of Kwatra [2] for the application of style transfer:

$$\frac{1}{c}\sum_{(i,j)\in\Omega_{L,n}}\min_{(k,l)}\|R_{ij}^{n}X-Q_{kl}^{n}S\|_{2}^{r}+\|X-C\|_{W}^{2}+\lambda\{X\} \quad (1)$$

Here:
X represents the estimated image
C represents the content image
S represents the style image
$R_{ij}^{n}$ represents extraction of the i,j-th patch of size nxn
$Q_{kl}^{n}$ represents extraction of the k,j-th patch of size nxn
$W$ represents a weight mask
$L$ represents the working scale
r & c represent regularization/normalization factors
$\lambda$ represents image prior statistics

Elad finds an algorithm that minimizes this cost function to produce an image that balances style and content according to a weight mask. Our approach builds off of this idea.

### C. Initialization

Some pre-processing and initialization is done on the input images (Figure 3) prior to running the algorithm.

Fig. 3: **Left**: Input content image. **Right**: Input style image

As a zeroth step, we assume (and ensure) for this work that all input images are 400x400 pixels, with 3 RGB color channels. However, this algorithm can be extended to images of variable dimensions and properties, though this is not discussed.

The first step is to apply a color transfer from the style image to the content image. This color transfer process is also used inside the algorithm and is described in Section III-H in more detail.

Next, the estimated image is initialized to the resulting content image plus strong Gaussian noise with zero mean and a standard deviation equal to the maximum value of the new content image (Figure 4).

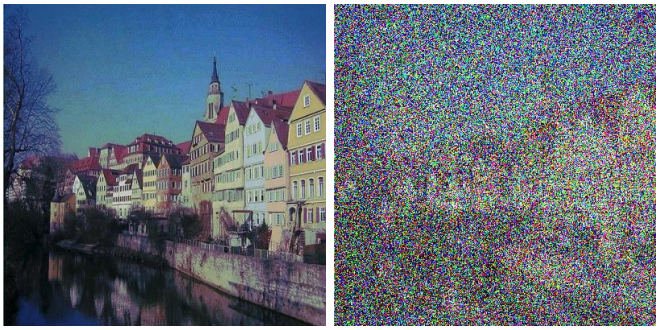

Fig. 4: **Left**: Example of strong additive Gaussian noise (see text) to 'house'. **Right**: Example of color transfer from 'Starry Night' to 'house'

The motivation for adding noise is to allow the patch matching step to make more venturous guesses. Indeed, not doing so will result in repeated patches (from the patch-matching step), especially in areas of high uniformity (which by definition are areas that contain similar patches) as shown in Figure 5.

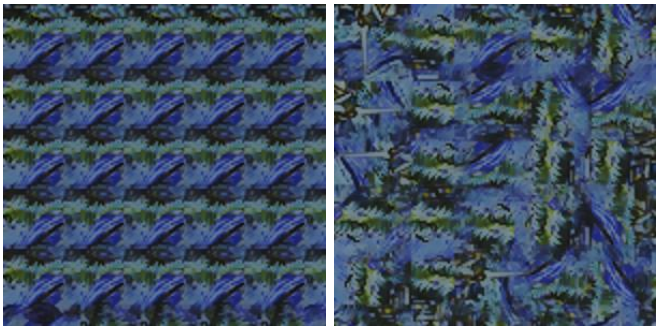

Fig. 5: **Left**: Example of patch matching on uniform white image with no noise **Right**: Example of patch matching on uniform image with added noise

Lastly, we generate a "hallucinated" image. This is an image that is supposed to represent purely the style, with little to no content. It is generated by running the described style transfer algorithm on a significantly blurred content image (Gaussian kernel, $\sigma = 100$), with a few minor alterations. Most notably, we skip the style fusion and content fusion steps. These steps do not apply since the style fusion depends on and uses the hallucination image; and we are not concerned with preserving the content while generating a hallucination. This image will be used later in the algorithm during the style fusion step.

### D. Style Fusion

In order to maximize the variability in the style transferred while still keeping true to the content, we propose applying

a weighted average between the estimated image and a hallucinated image in each iteration. The weighting for the two images can be adjusted as desired. For our implementation, we selected relatively conservative values of 75% and 25% for the estimated output and hallucinated images, respectively.

Fig. 6: Example of style fusion applied to 'Starry Night' style image and 'house' content image, noise initialization not shown for clarity **Left:** Hallucinated image **Right:** Resulting image

### *E. Patch Matching*

The objective of patch matching is to extract areas of the style image that mimic areas of the content image. To do this, we first choose a patch in the content image that we want to find a style-match for. Then, we perform a nearest-neighbor (NN) search in the style image : we iterate through patches in the style image and select the patch which has the lowest $L_2$-norm with respect to the content patch. The NN search can be formulated as the following optimization problem,

$$\{k^*, l^*\} = \underset{(k,l)}{\text{Argmin}} \, \|R_{ij}^n X - Q_{kl}^n S\|_2 \tag{2}$$

This process is computationally intense and it is optimized in two ways.

First, we note that considering every possible patch of each image (style and content) does not improve the quality of the output image over skipping some patches. The reason is that consecutive patches are similar to each other, so only one patch out of a certain area is needed to contribute useful information about the style of that area. Thus, we chose to introduce a regular grid of overlapping patches from which to choose our content and style patches. The reduction in computation is proportional to the square of 1 + however many pixels are skipped; in our case, we achieve, on average, a 99.4% reduction. Since we were able to reduce the patch space by a great margin without greatly compromising fidelity, we decided to also include the three other major rotations of each patch. Doing so gives the matching algorithm a more diverse distribution of style patches to sample from, which may decrease the minimal $L_2$-norm of the selected style patch in some cases.

Patch matching was also optimized with principal component analysis (PCA). By projecting every possible patch of the style image into a smaller space before computing the norm, we greatly decrease the computational effort of the minimization (with the trade-off of using more memory). The basis (and dimension) of this projection space is chosen from the eigen-vectors whose energy contribution is 95% of the original total energy of the eigen-space of patches. The remaining eigen-vectors do not store critical information in differentiating style patches from each other, so they are not considered. By performing PCA, we are able to decrease the dimension of the original style patch space, on average, by 85%.

After calculating the $L_2$-norm of each style patch to the content patch, we add noise to each calculated metric associated with a patch to allow the matching algorithm to make venturous guesses during run-time. The Gaussian noise we added had a standard deviation of 10% of what the minimum $L_2$-norm would have been. This level of noise was designed so that we would choose a style patch from a similar range of $L_2$-norms.

Finally, we report the original patch in the style image that corresponds to the chosen projected patch as most similar to the given content patch (i,j) based on the $L_2$-norm metric

$$z_{ij} = Q_{k^*,l^*}^n S \tag{3}$$

### *F. Style Synthesis*

We now attempt to use the style patches, found using NN, to estimate the output image. In essence, each $i, j$ patch in the output image should roughly match the corresponding NN style patch, $z_{ij}$. This can be formulated as the following optimization problem,

$$\widetilde{X} = \underset{X}{\text{Argmin}} \sum_{(i,j)\in\Omega} \|R_{ij}^n X - z_{ij}\|_2^r, \tag{4}$$

which can be directly approximated using iterative re-weighted least squares (IRLS), as used by Kwatra [2].

### *G. Content Fusion*

To ensure that the original content is preserved, we use a pixel-by-pixel weighted average of the content image and the estimated style image. In areas where there is more "content," the weight will be heavier on the content image. In areas where there is less "content," the weight will favor the estimated image. The pixel-by-pixel weight mask is calculated by performing a series of segmentation algorithms on the content image.

First, because the content is usually in the foreground of the image, we assign the background of the content image a small weight and the foreground a larger weight. The foreground / background separation is achieved using active contour based segmentation.

Next, because the edges of the content image contain the most information in an image, we attribute a larger weight to the edges of the content image. These edges were extracted from the zero crossings of the Laplacian of Gaussian operation. Lastly, to make transitions between high content areas and high-style areas smoother, we apply a Gaussian smoothing to the mask. An example mask is shown in Figure 7.

Equation 5 is used to apply the content fusion and update the estimate.

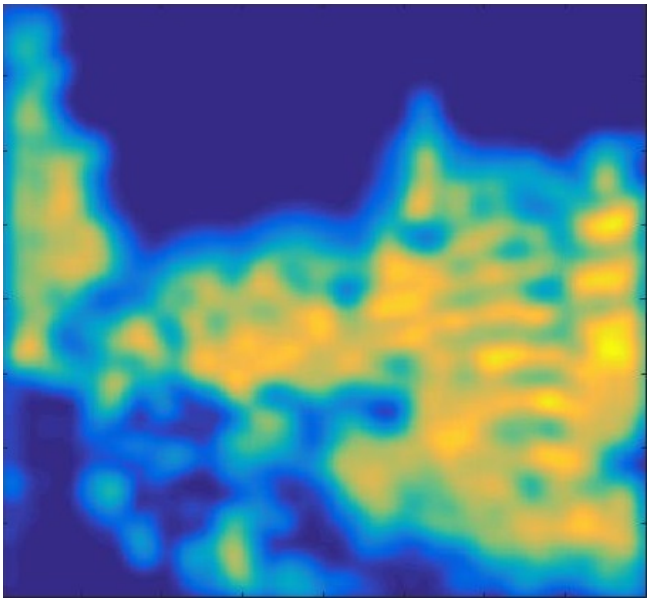

Fig. 7: Example mask generated from "house" image

$$\widehat{X} = (W + I)^{-1}(\widetilde{X} + WC) \tag{5}$$

Fig. 8: **Left**: Example $\widetilde{X}$, the style synthesis output before content fusion. **Right**: Example $\widehat{X}$ after content fusion using above mask

### *H. Color Transfer*

The color palette of the final output image can be chosen to be anything desired. For example, we could either keep the color palette of the original content image or take the colors from the style image. Since, the colors used in the style image can be interpreted as part of the style, we chose the latter approach. This also results in more visually pleasing results since there is generally more diversity in the color palette of the style image. In our implementation, we use the MATLAB function, **imhistmatch**.

### *I. Denoising*

The last step of the inner-most loop is to restore our image, which has been heavily processed, to the prior statistics of a general image. Indeed, we see artifacts of the patch selection grid and content fusion edges in the output image. To mitigate these effects, we choose a filter that smooths weak edges and preserves strong ones - a bilateral filter, specifically the Domain Transfer Filter [3].

### *J. Scale Space*

In order to achieve a more diverse and robust style transfer, the above procedure is applied to multiple resolution scales. For a given patch size, the larger the scale (i.e. lower resolution) we use, the larger the extracted features will be. As discussed in [1], the various scales can be combined into a single optimization problem. However, for simplicity, we sweep over the scales sequentially, from coarsest to finest.

### *K. Patch size*

Another way we can influence the scale of the style transfer is by choosing different patch sizes. This has the added benefit of smoothing the resulting patch grid artifacts as smaller patches are designed to lie across the boundaries of the larger ones. Additionally, by choosing different patch sizes, we increase the variability in the distribution of patch choices and increase the refinement of details in the style transfer.

In general, varying both the scale space and patch size will offer the algorithm a diverse array of structures from different scales and proportions from which it can sample for patch matching.

## IV. Results

This section describes the results we obtain with the algorithm, including a brief discussion of failure cases and parameter selection.

### *A. Examples*

Our algorithm obtains fairly robust style transfer for a variety of different style and content images. A selected subset is shown in Figure 9.

### *B. Failure Cases*

This section will discuss conditions in which we have seen our algorithm perform poorly and possible next steps for mitigation.

*1) Incompatible Style Image:* One case where our algorithm may not produce pleasing results is when the style image does not provide a good representation of the style. This of course is not well defined, but we have found that style images that do not posses many colors or only have a small region of desired style generally produce poorer results. Figure 10 shows an example of this, where the style image is largely composed of just two hues and most of the detail is located in the horse's head, which covers only a fraction of the image. As shown in the resulting hallucinated and estimated image, there is a lack of variety and results in mainly blotches of the two colors present in the style image.

Another issue with certain style images is that if the style inherently contains a lot of detail, PCA will not optimize it as well as other styles. For example, in Starry Night, there is a lot of fine high frequency variation seen in the sky that

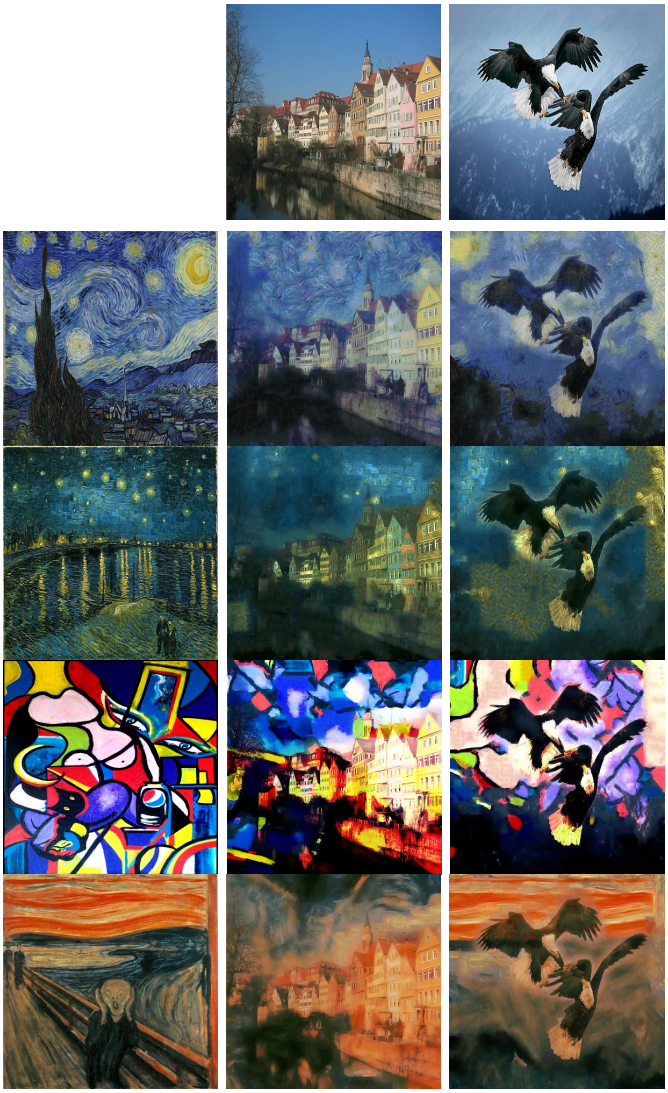

Fig. 9: **Top:** Content images, 'house' and 'eagles' **Left:** Style images **Middle:** Style transfer results with 'house' **Right:** Style transfer results with 'eagles'

contributes to the style of the image. On the other hand, in the case of Der Schrei (Scream), the style is smoother and has less fine detail. Thus, we see close to a 2x computation speed increase for the Der Schrei style image as compared to Starry Night.

Fig. 10: **Left:** Style image **Middle:** Hallucinated image **Right:** Estimated image

*2) Incompatible Content Image:* The most common failure mode for incompatible content images is when our segmentation does not successfully detect areas of relevant content. Indeed, Elad details his encounter with the same problem [1]. For example, in images of human faces, our segmentation algorithm interprets all smooth and uniform areas as content-poor. Thus, the forehead and cheeks are assigned a low weight in the mask, despite our intuition that they are key features of the face, an area of high content. Eventually, because of their low mask weight, these areas are populated with more style. The juxtaposition of these highly styled areas against other content-rich areas of the face with less style (such as the eyes and mouth) is aesthetically strange as our minds are conditioned to perceive the whole face as one entity. To mitigate this, Elad has shown that it is possible to alter the segmentation mask to get good results if the algorithm has prior knowledge that the content image contains a face [1].

Fig. 11: **Left:** Mask for selfie image - note the blue (nominally lower) areas of the forehead and cheeks **Right:** Final image using segmentation mask on left

### C. Parameter Selection Analysis

This section details our design of the parameter space and how each parameter influences the resulting image

*1) Patch Sizes:* For the final algorithm, we have chosen the patch sizes: 36x36, 22x22, and 13x13. The sizes of the patches were designed so that we would get a broad spectrum of structures from the style image while not compromising on speed and memory usage. In general, decreasing a patch size will increase the computation time because there will be more matches to consider for NN. Additionally, with more patches, we can eliminate grid artifacts that occur from the choice of our patch shape. These artifacts are reduced because each consecutive patch size was designed so that a patch would exist above the previous patch's grid line. An example demonstrating the effect of different patch sizes is shown in Figure 12.

*2) Scales:* Here we experiment with a combination of four different scales ($L = 1, 2, 4, 8$), where a scale of $L$ means that the images are re-sized by a factor of $1/L$. Looking at Figure 13, we see that larger structures start appearing when we reach a scale of $L_{max} = 4$. However, using a larger scale, $L_{max} = 8$, does not seem to offer much improvement. Therefore, in our implementation we use scales $L = 1, 2, 4$. See Section III-J for more details about how scale space is used in the algorithm.

Fig. 12: **Left**: [36x36] patch size **Center:** [36x36,22x22] patch sizes **Right:** [36x36,22x22,13x13] patch sizes

Fig. 13: Example output images using different number of scales. **Top-Left:** $L = 1$ **Top-Right:** $L = 1, 2$ **Bottom-Left:** $L = 1, 2, 4$ **Bottom-Right:** $L = 1, 2, 4, 8$

*3) NN Noise:* /Figure 14 demonstrates the effect of the noise added in the NN search, which is essentially an approximate NN.

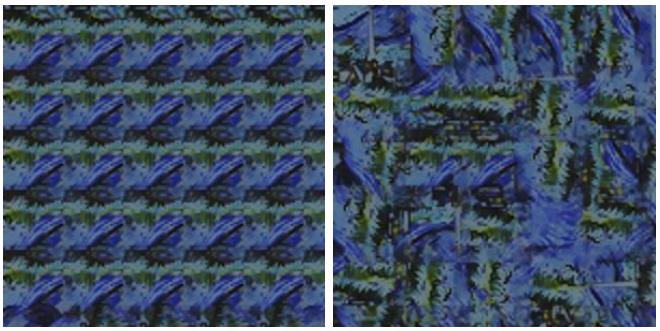

Fig. 14: **Left:** Example of patch matching on uniform white image with no noise **Right:** Example of patch matching on uniform image with 10% NN noise, see Section III-E

This little addition of variability makes each generated image unique, even when given the same input content and style images. This is demonstrated with hallucinated images in Figure 15.

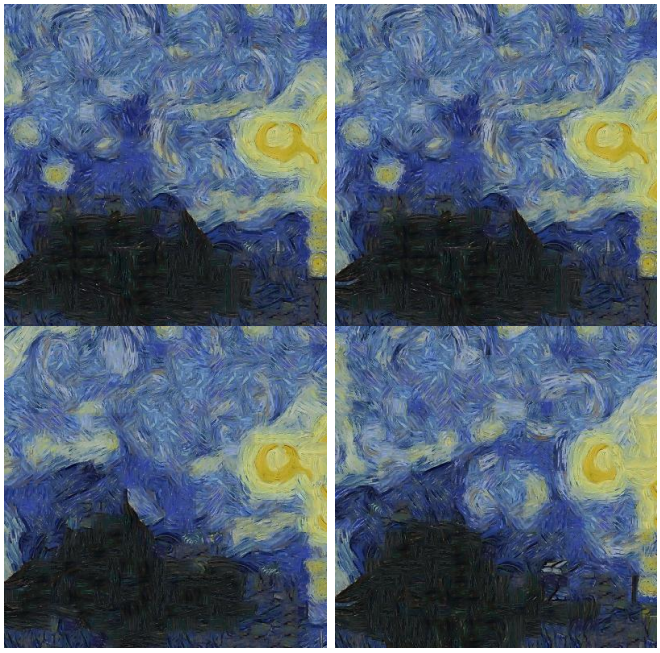

Fig. 15: **Top:** Example of hallucinated images with no NN noise **Bottom:** Example of hallucinated images with 10% NN noise

*4) Hallucination Weight:* By varying how much weight the Hallucination has on the estimated image, we change how much variation in style we see in the final output. For example, with a hallucination weight of 0%, we see more random styles being generated. Specifically, in the sky, we can see yellow streaks that bend at angles not necessarily found in starry night. However, with a hallucination weight of 50%, we see style that has a better match with the broader swirling structure of starry night. The reason for this is that, in the 0% hallucination case, the uniform areas are confined by the boundary of the mask. This will limit the types of structures that can exist in that area. On the other hand, using a greater percentage of hallucination will perpetuate the larger structures that are generated in the hallucination image, which, at the time of its generation, was not limited by a mask (it skips the content fusion step). To balance these two effects, we choose a hallucination weight in the middle for our final algorithm. We found that a weighting of 25% hallucination allows the algorithm to generate its own style, while still maintaining the broader structures of the style.

## V. Conclusion

Our algorithm demonstrated results of similar quality to those using CNNs in the work of Gatys [4], while using a more image processing style approach. Conceptually, we use ideas from Elad and Milanfar [1] for achieving style transfer through the texture synthesis results of Kwatra et al. [2].The main additions of our work are the inclusion of the hallucinated image in the algorithm pipeline as well as the use of rotated patches in the NN patch matching step. These allow us to gain more variability and uniqueness in the images we produce, a generally desired attribute. However, as noted in [1], what constitutes a successful style transfer is not definitive. And

Fig. 16: **Top Left**: 0% hallucination weight, **Top Right**: 10% hallucination weight, **Bottom Left**: 25% hallucination weight, **Bottom Right**: 50% hallucination weight

therefore, users may wish to tune the parameters differently to suit their preferences.

### A. Future Directions

One clear area of improvement is making the algorithm robust to a greater range of style and content images. This can be done by defining regions of the style image with a richer selection of style and also having a better segmentation process or removing the need for a mask in the algorithm. While computational speed was not a primary concern during the development of our algorithm, admittedly it can take from approximately 10-20 minutes to generate an image (or about half as long if a hallucinated image is provided or not used). However, a combination of MATLAB and algorithmic optimizations should be able to reduce the run-time significantly.

## APPENDIX A
## CODE

The GitHub repository containing the code can be found here: https://github.com/ewang314/EE368_Final_Project

## ACKNOWLEDGMENT

We thank Professor Gordon Wetzstein for guidance in the initial project formulation.

## REFERENCES

[1] M. Elad, P. Milanfar, *Style-Transfer via Texture-Synthesis*, Google Research, September 21, 2016

[2] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, *Texture Optimization for Example-Based Synthesis*,ACM ToG, Vol. 24, No. 3, pp. 795-802, 2005

[3] E.S.L. Gastal and M.M. Oliveira, *Domain Transform for Edge-Aware Image and Video Processing*, TOG, Vol. 30, No. 4, pp. 169, 2011.

[4] L.A. Gatys, A.S. Ecker, and M. Bethge, *Image Style Transfer Using Convolutional Neural Networks*, CVPR, 2016.

[5] Zhang, Wei, Chen Cao, Shifeng Chen, Jianzhuang Liu, and Xiaoou Tang. *Style transfer via image component analysis* IEEE Transactions on Multimedia 15, no. 7 (2013): 1594-1601