CS 445 Project: Painting Without Paint

Members
Joe Morrissey (jrm15), Kenny Kim (kk67), Madhuparna Bhowmik (bhowmik6), Sahil Agrawal (sahila4)

Motivation
We found it fascinating that we could mimic an artist's style and create art that could have potentially been done by them. So if we have a photo of an object and we want to see how an artist like Manet, Leonardo da Vinci, etc. would have painted it, we can use this technique to do so. Additionally, Image Analogies can be used to perform tasks like texture transfer, cartoonify an image, super-resolution, etc. Combining this with Color Transfer opens up the possibility of not only mimicking the style of an image in another, but also having the same range of colors. This can be used in photography editing like making a day scene into a night or sunset photo.

Approach

1. **Image Analogy**

The purpose of image analogy is to use an input image and its filtered pair as training data to create a new "analogous" image with similar textures and visual looks to the original. This allows us to model any filter without knowing how the filter was originally applied. To achieve our results in image analogies, we first have to set up the gaussian pyramids of the luminance channel for three images, A, A' and B. A is the original image, A' is the original image with the unknown filter applied, and B is the "analogous" image we want to apply the filter to. In each level of the gaussian pyramid, we iterate through each pixel of an empty B', the final output image, and determine the best pixel to fill in.

We used two separate functions to determine the best pixel. The first is BestCoherenceMatch, where we use neighborhood matching to find the best matching pixel from A. Neighborhood matching works by taking into account the already synthesized pixels of B', the corresponding pixels of A', a 5 x 5 neighborhood in A and B, and a 3x3 neighborhood in the coarser levels of A, A', B, and B'. We decided to use the luminance channel for our feature vectors because it allowed for a speedup in finding B', since we have to store one third of the channels. The second is BestApproximateMatch, where we used an approximate-nearest-neighbor (ANN) search to determine the closest pixel in the current level of the gaussian pyramid. In our implementation, we chose to use a KDtree from the sci-kit learn library to query through the feature vectors and find our approximate pixel. Once the BestApproximateMatch and BestCoherenceMatch returned results, we compared the error for each pixel and chose the minimum of the two. Pseudocode Shown Below.

function BESTMATCH(A, A', B, B', s, ℓ, q):
    $p_{\text{app}} \leftarrow$ BESTAPPROXIMATEMATCH(A, A', B, B', ℓ, q)
    $p_{\text{coh}} \leftarrow$ BESTCOHERENCEMATCH(A, A', B, B', s, ℓ, q)
    $d_{\text{app}} \leftarrow \|F_\ell(p_{\text{app}}) - F_\ell(q)\|^2$
    $d_{\text{coh}} \leftarrow \|F_\ell(p_{\text{coh}}) - F_\ell(q)\|^2$
    if $d_{\text{coh}} \leq d_{\text{app}}(1 + 2^{\ell - L}\kappa)$ **then**
        return p_{coh}
    else
        return p_{app}

After looping through every level, we then took the luminance channel from the finest level of the pyramid and copied over the I and Q channels from the original B image. This works since our eyes are more sensitive to luminance channel changes than to color difference channels changes.

We had to make some design choices where the paper was vague. Initially instead of doing a brute force choice at the coarsest level and border of the image, we passed in an expected result from the paper to fill in pixels at the border. We also didn't loop through the coarsest image in CreateImageAnalogy. This allowed for faster runtimes and to produce more results, although we finally managed to implement the brute force search as well which makes the method independent of the expected result. All our results are using this method.

2. Color Transfer

To perform color transfer we extracted the mean ($[\overline{R}_{src}, \overline{G}_{src}, \overline{B}_{src}]$, $[\overline{R}_{tgt}, \overline{G}_{tgt}, \overline{B}_{tgt}]$) and covariance ($Cov_{src}$, Cov_{tgt}) of each channel of the source and target images. Then we decomposed the covariance matrix using SVD (Singular Value Decomposition) to get the two orthogonal matrices U, V and the diagonal matrix Λ, with eigenvalues.

$$\mathbf{Cov = U \cdot \Lambda \cdot V^T}$$

U was used as the rotation matrix and we used the eigenvalues for the source matrix. The final formula used to generate the color-transferred image is as follows:

$$\mathbf{I_{result} = T_{src} \cdot R_{src} \cdot S_{src} \cdot S_{tgt} \cdot R_{tgt} \cdot T_{tgt} \cdot I_{tgt}}$$

Here, I_{tgt} is the target image and the T,R and S matrices are as follows:

$$T_{src} = \begin{bmatrix} 1 & 0 & 0 & \overline{R}_{src} \\ 0 & 1 & 0 & \overline{G}_{src} \\ 0 & 0 & 1 & \overline{B}_{src} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{tgt} = \begin{bmatrix} 1 & 0 & 0 & -\overline{R}_{tgt} \\ 0 & 1 & 0 & -\overline{G}_{tgt} \\ 0 & 0 & 1 & -\overline{B}_{tgt} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad S_{src} = \begin{bmatrix} \sqrt{\lambda}^{r}_{src} & 0 & 0 & 0 \\ 0 & \sqrt{\lambda}^{g}_{src} & 0 & 0 \\ 0 & 0 & \sqrt{\lambda}^{b}_{src} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

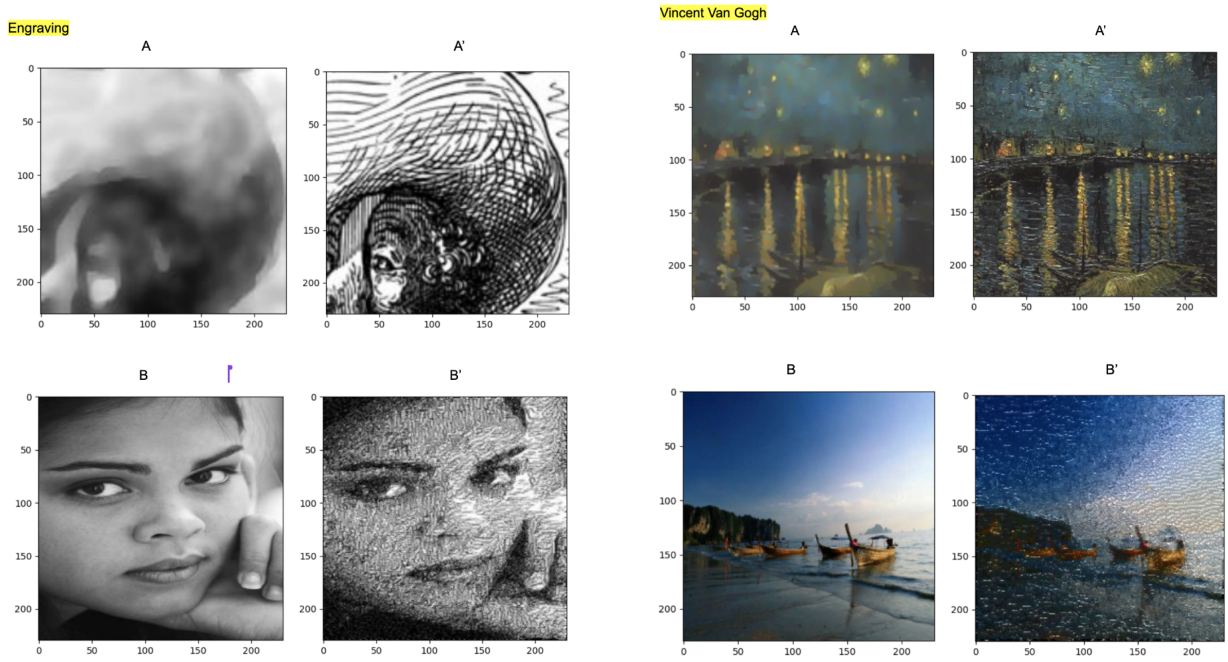
$$S_{tgt} = \begin{bmatrix} 1/\sqrt{\lambda}^{r}_{tgt} & 0 & 0 & 0 \\ 0 & 1/\sqrt{\lambda}^{g}_{src} & 0 & 0 \\ 0 & 0 & 1/\sqrt{\lambda}^{b}_{src} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_{src} = U_{src} \quad R_{tgt} = U^{-1}_{tgt}$$

For swatch-based color transfer, we divided the source and target image into the same number of pieces with each piece of the source transferring its color to a matching piece in the target image. In this way, we were able to have different parts of the target image get its color from a different part of the source image.

Result

1. Image Analogy

Examples of image analogy applied on engravings and a Van Gogh style painting. Clearly, the style of the engraving and Van Gogh painting have been transferred to the normal image and it looks like the final image was engraved or painted by Van Gogh. Our results are similar to the expected result. The results shown here are from papers, we applied this on a new set of images, the results of which are in the linked document.



Expected Results from Paper:

Link to more results: **Image Analogies Results**

2. Color Transfer

We applied color transfer on a scene of a kitchen with yellow lighting to be transferred to a bathroom scene with natural lighting and the resulting image has the color scheme of the source image, that is, yellow lighting. In swatch-based color transfer, we were able to transfer the color of one monument (Statue of Liberty) onto another monument (India Gate) which makes it feel like the India Gate is made of green copper which is not actually the case.

Source Image	Target Image	Result Image	Expected Output

Swatch-based Color Transfer

Source Image	Target Image	Source Mask	Target Mask	Result
				

Link to more Results: Color Transfer Result

Implementation Details

1. **Image Analogy**
   We used cv2, matplotlib, numpy, and sci-kit for our implementation. To create our own input images for image analogies, we used the utils from Project 1 in order to obtain a gaussian blurred filter image. We also used the KDtree from the sci-kit learn library to aid with Approximate Nearest Neighbor search in our BestApproximateMatch. We used 4 levels of pyramids and a neighborhood of 5X5 for the same level and 3X3 for the lower level.
2. **Color Transfer**
   We used the cv2, matplotlib, numpy and skimage libraries in Python to implement Color Transfer. We read an RGB image as input and created the 6 matrices described above. We had to pad the U matrix with zeros before converting it into the rotation matrix, R. We also converted the target image to its homogeneous coordinates before calculating the final result image. The implementation of swatch-based color transfer required finding the mask for different swatches. To achieve this we used the utils from Project 3. We selected points around the swatch to get the polygon boundaries and then get a mask from this. Then we ran the Color Transfer algorithm only on these swatches individually and combined the results to get the final Color Transfer result.

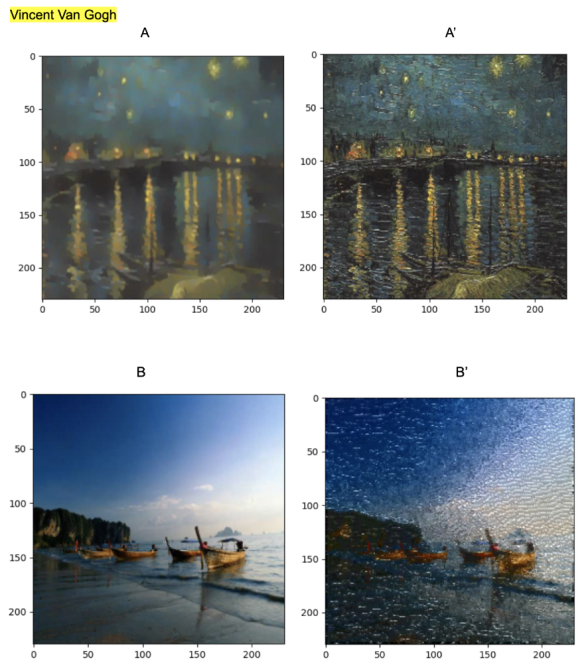
Challenge/Innovation

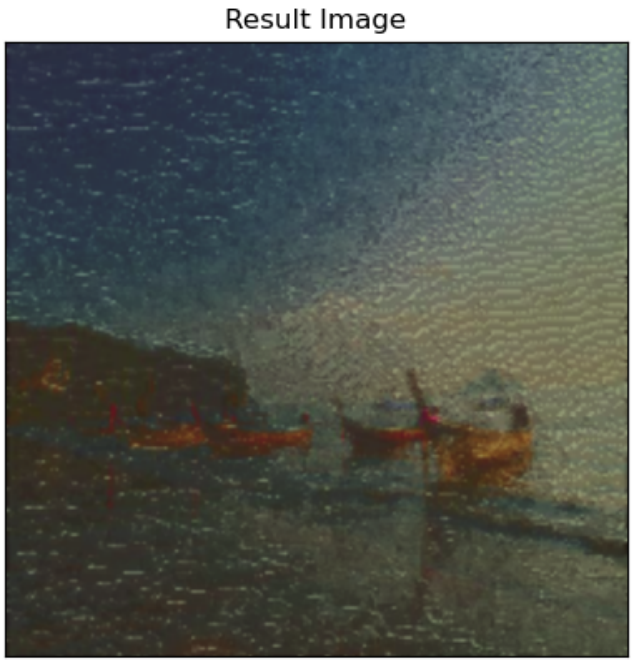
We expect to receive 20 points for the challenge/innovation section as we have successfully overcome all the challenges mentioned below and also extended an existing technique by combining Image Analogies and Color Transfer and extending Color Transfer to video sequences as well as mentioned below.

1. **Challenges**
   a. There was a mistake in the equation in the Color Transfer Paper which was leading to bad results. We had to spend some time trying to figure out the issue. The values for the matrix S_{src} , specifically s^r_{src} ,s^g_{src} , and s^b_{src} were wrong and we had to experiment and find the correct values.
   b. In the Image Analogy paper lots of the implementation details were vague. For e.g. they did not specify how to get the features for corner and edge pixels and perform brute force search on such pixels.
   c. Additionally, in the Image Analogy paper, the authors say they normalize the vectors so each scale of the pyramid has equal weight, but they don't mention which vector to normalize and when.
   d. Another small challenge is that Image Analogy takes 20-30 minutes to run due to the brute force search for edge pixels and so it was hard to debug and test the method.

2. Innovation 1: Image Analogy + Color Transfer

To mimic an artist's style using only Image Analogy or Color transfer is not sufficient. Thus, by combining the two we can more accurately mimic an artist's painting style. Hence we combined the two approaches to get the following results.





3. Innovation 2: Color Transfer on a Video

To perform swatch-based color transfer on a video sequence it is required to know the mask for an object in each frame. However, manually specifying a mask for each frame is time-consuming. Thus, we used ideas from Project 5 to find transformation of the object in the nearby frames. For this we used the auto homography function and applied the transformation on the mask of the reference frame which is the middle frame in the video sequence. In this way, we can specify the mask for just the reference frame and it will automatically be transformed for all the frames provided the masked object is still visible.
In the video, the camera moves around so the phone rotates, translates and gets scales in the video and the mask is automatically adjusting and the color of the phone is changed from pink to blue and the table from brown to multicolored. (video link)

Resources/References

- **[1]http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15463-f08/www/final_proj/www/hstrong/**
- **[2]https://mrl.cs.nyu.edu/publications/image-analogies/**
- **[3]https://dl.acm.org/doi/10.1145/1128923.1128974**
- **[4]https://ieeexplore.ieee.org/abstract/document/946629**
- **[5]numpy - Nearest Neighbor Search: Python - Stack Overflow**