

---

# FAST GPU-ENABLED COLOR NORMALIZATION FOR DIGITAL PATHOLOGY

---

A PREPRINT

**Goutham Ramakrishnan\***

Department of Electrical Engineering  
Indian Institute of Technology Bombay  
Mumbai, India  
gouthamr@iitb.ac.in

**Deepak Anand**

Department of Electrical Engineering  
Indian Institute of Technology Bombay  
Mumbai, India  
deepakanand@iitb.ac.in

**Amit Sethi**

Department of Electrical Engineering  
Indian Institute of Technology Bombay  
Mumbai, India  
asethi@iitb.ac.in

January 11, 2019

## ABSTRACT

Normalizing unwanted color variations due to differences in staining processes and scanner responses has been shown to aid machine learning in computational pathology. Of the several popular techniques for color normalization, structure preserving color normalization (SPCN) is well-motivated, convincingly tested, and published with its code base. However, SPCN makes occasional errors in color basis estimation leading to artifacts such as swapping the color basis vectors between stains or giving a colored tinge to the background with no tissue. We made several algorithmic improvements to remove these artifacts. Additionally, the original SPCN code is not readily usable on gigapixel whole slide images (WSIs) due to long run times, use of proprietary software platform and libraries, and its inability to automatically handle WSIs. We completely rewrote the software such that it can automatically handle images of any size in popular WSI formats. Our software utilizes GPU-acceleration and open-source libraries that are becoming ubiquitous with the advent of deep learning. We also made several other small improvements and achieved a multifold overall speedup on gigapixel images. Our algorithm and software is usable right out-of-the-box by the computational pathology community.

## 1 Introduction

Tissue samples stained with any stain in general, and hematoxylin and eosin (H&E) in particular, suffer from variability in their appearances (see Figure 1), which arise due to differences in the staining protocols and reagents used to process them. In digital pathology, the sensor response of the scanners used to capture their images can also add to this variability. While human visual perception automatically adjusts to differences in stain appearances, the performance of machine learning and deep learning algorithms that analyze these images depends on seeing enough variation in the training data. Conversely, the performance of these algorithms improves with color normalization Sethi et al. [2016]. This makes color normalization a necessity when the data is sourced from only a few labs.

Several color normalization algorithms for histological images have been recently proposed, of which Khan *et al.* Khan et al. [2014], Macenko *et al.* Macenko et al. [2009] and Reinhard *et al.* Reinhard et al. [2001] are the most popular. Vahadane *et al.* proposed a technique for stain separation and color normalization called structure preserving color

---

\*Equal contributions by first two authors

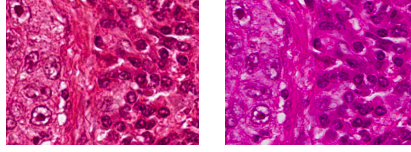


Figure 1: **Need for color normalization:** The same tissue slide scanned with Aperio and Hamamatsu scanners respectively Vahadane et al. [2016]

normalization (SPCN), and released its source code Vahadane et al. [2016]. They demonstrated that it performed qualitatively better in preserving biological structure, and quantitatively better in preserving stain densities compared to the previously popular techniques. However, no software exists for public use that can handle large gigapixel images that are common in digital pathology. Additionally, as well-motivated as SPCN is, it occasionally produces undesirable artifacts.

In this work, we significantly improve upon SPCN and its prior implementation to introduce a software for color normalization that (a) is free for non-commercial public use based on open-source software, (b) can handle large images without the need for embedding it in custom code to break up images, (c) scales gracefully in its runtime with image size by utilizing GPU acceleration, and (d) avoids undesirable artifacts that are common in the results of SPCN.

## 2 SPCN Algorithm and its Shortcomings

We chose to improve and scale up the SPCN algorithm because it is intuitively satisfying in how it handles histological images. It first breaks a source image down into its stain densities, and then normalizes the color basis for each stain with respect to those of a target image. Additionally, the processes of stain density estimation and color normalization themselves are well-motivated. In this section, we revisit SPCN and document the strengths and shortcomings of its core algorithm and Matlab<sup>®</sup> implementation.

### 2.1 SPCN Algorithm

Stain separation process in SPCN is based on the following insights about the staining process Vahadane et al. [2016]: (a) stain densities that modulate optical densities are non-negative, (b) their mixing proportions and color components are also non-negative, and (c) each pixel is likely to represent a spatially contiguous biological structure that absorbs largely one stain.

The major steps of SPCN based on these insights are given below Vahadane et al. [2016]:

1. **Optical density computation:** The insight about the need to model the optical densities as opposed to the pixel intensities motivated the use of Beer-Lambert transform. Relative optical density  $v_{c,x,s}$  corresponding to intensity  $i_{c,x,s}$  for each channel  $c \in \{red, green, blue\}$  for each pixel location  $x$  for each image  $s \in \{source, target\}$  is computed by assuming a maximum intensity  $i_0$  (fixed to 255 for 8-bit images in the original implementation) corresponding to zero optical density and by using Beer-Lambert law as follows:

$$v_{c,x,s} = \log \left( \frac{i_0}{i_{c,x,s}} \right) \quad (1)$$

2. **Unsupervised stain density estimation:** The non-negativity of optical densities, stain densities, and their mixing proportions motivated the use of non-negative matrix factorization (NMF). Stain specificity of each biological structure motivates the use of sparse NMF (SNMF) to perform soft clustering of each pixel into stain clusters. The relative optical density  $V_s$  (which is a matrix of  $v_{c,x,s}$ ) can be decomposed into its constituent non-negative factors – the color basis matrix  $W_s$  and the stain density matrix  $H_s$  such that  $V_s \approx W_s H_s$ . To obtain this decomposition we solve the following optimization problem assuming that the rank  $r$  of  $H_s$  is two (number of stains) for H&E:

$$\min_{W_s, H_s} \|V_s - W_s H_s\|_F^2 + \lambda \sum_{j=1}^r \|H_s(j, :)\|_1, \quad W_s \geq 0, H_s \geq 0, \|W_s(:, j)\|_2^2 = 1 \quad (2)$$

The penalty on the  $L1$  of  $H_s$  induces sparsity, which is controlled by the hyperparameter  $\lambda \geq 0$  (fixed to 0.1 as per Vahadane et al. [2016]). Dictionary learning is used to estimate  $W_s$  and its pseudo inverse is used to compute  $H_s$ .

3. **Color normalization:** After stain separation the source image is normalized with respect to the target image by replacing the former’s color basis with that of the latter while preserving the former’s relative stain densities. The source stain density for each stain is linearly scaled so that its 99<sup>th</sup>-percentile value matches that of the corresponding target stain. That is, the normalized matrix  $V'_{\text{source}}$  is represented as:

$$V'_{\text{source}} = \frac{P_{99}(H_{\text{target}})}{P_{99}(H_{\text{source}})} W_{\text{target}} H_{\text{source}} \quad (3)$$

4. **Normalized pixel intensity computation:** The source optical densities for each channel and each pixel thus normalized with respect to the target image are then converted back to the pixel intensity space using the inverse Beer-Lambert transform as follows:

$$i'_{c,x,\text{source}} = i_0 \exp(-v'_{c,x,\text{source}}) \quad (4)$$

## 2.2 Occasional qualitative defects produced by SPCN

In our experimentation with SPCN, we observed the following problems:

1. **Invalid maximum channel intensity assumption:** SPCN assumes a fixed maximum intensity value  $i_0$  for each color channel in equations 1 and 4. When the background area of a slide has a color tinge or is slightly darker than the maximum possible brightness (e.g. 255), this assumption is no longer valid. This occasionally leads to undesirable color artifacts in the normalized image.
2. **Inconsistency in extracting stain color prototype:** When there is lack of density variation in one stain due to a few very dark nuclei or a large proportion of light background, the optimization problem in equation 2 can become ill-conditioned. This problem is exacerbated in SPCN as it cannot process large images efficiently.
3. **Unintentional stain swapping:** While solving the optimization problem in equation 2, the order of the two color bases (columns of  $W_s$ ) and consequently stain density estimates (rows of  $H_s$ ) can get swapped between the two stains.

## 2.3 Challenges in applying SPCN to WSIs

To its credit, a fast approximate scheme for estimating  $W_s$  in equation 2 was proposed for SPCN in Vahadane et al. [2016], but it is still not a software that can be used out-of-the-box for WSIs. The Matlab<sup>®</sup> implementation of SPCN does not have support to read WSI formats such as *.svs*, *.ndpi* (from popular scanner brands such as Aperio and Hamamatsu) to begin with, which is a platform integration issue. Additionally, it reads the entire source or target image in one go for estimating  $H_s$  in step 2, which leads to program crashes for WSIs that cannot fit in the RAM. It does not utilize GPUs for computationally heavy tasks such as computing Beer-Lambert transform, its inverse, or normalizing each pixel in equations 1, 4, and 3 respectively, even though GPUs are becoming commonplace due to the increasing application of deep learning to pathology. Using SPCN in a loop over patches in a compatible format extracted from a WSI is problematic, because it will simply normalize each patch independently, leading to color inconsistency over a WSI.

## 3 Proposed Algorithm and its Implementation

We now describe key features of our software<sup>2</sup>.

### 3.1 Improving qualitative results

We solved the problems mentioned in Section 2.2 as follows:

1. **Neutral background color:** To fix the background tinge, we replaced the fixed maximum  $i_0$  of equations 1 and 4 that deal with Beer-Lambert transform by channel-wise and image-wise maximum  $\max_x(i_{c,x,s})$ . For outlier rejection while maintaining efficiency for WSIs, we sample at most 100,000 pixels with intensity above 220, and choose their 80<sup>th</sup>-percentile as our estimate for the maximum channel intensities.
2. **Color consistency over a WSI:** Although we seamlessly process WSIs of any size patch-wise, we estimate the color basis only once per WSI to maintain color consistency. This was not possible with SPCN because it

<sup>2</sup>Software available at: [https://github.com/MEDAL-IITB/Fast\\_WSI\\_Color\\_Norm](https://github.com/MEDAL-IITB/Fast_WSI_Color_Norm)

can only treat patches of a WSI independently. We use large enough samples of non-white pixels (intensity less than 220) from the entire WSI to increase the variance of different stains observed, which prevents equation 2 from becoming ill-conditioned while estimating  $W_s$ . At the same time, because we sub-sample a gigapixel image, we solve the optimization problem efficiently. Additionally, while scaling using the 99<sup>th</sup>-percentile values attempts to reject dark outlier pixels while normalizing stain densities, this can still lead to problems in images with a large background portions. We use the 99<sup>th</sup>-percentile of the sub-sampled non-white pixels to reject problems of both dark outliers and white pixels.

3. **Unintentional stain swapping:** The original heuristic implemented to get the stain order correct only compared the average blue channel of each stain. This leads to problems when the epithelium is light in color or the stroma also has a blue tinge. We resolved this issue by comparing the difference between red and blue channels across the two stains to get hematoxylin and eosin in the correct order.

### 3.2 Improving efficiency and WSI compatibility

Our next set of contributions are in the form of a complete re-implementation of SPCN for efficiency and scale up to handle WSIs, as noted below:

1. **WSI format compatibility using open-source software:** We implemented SPCN in python, which allowed us to use OpenSlide library Goode et al. [2013] for reading popular WSI file formats such as *.svs* and *.ndpi*.
2. **Efficient estimation of color basis:** As mentioned in Section 3.1, computation of  $W_s$  can easily be scaled up by limiting the number of pixels used for stain color basis estimation, for which we use the open-source SPAMS library Mairal et al. [2010]. Our software attempts to find a sufficient number of randomly scattered non-white pixels by finding at most 20 non-background patches on which the optimization problem of equation 2 is solved. This step thus scales sub-linearly with WSI size and is similar to the speedup scheme suggested in Vahadane et al. [2016]. It isn't constant because the search space for a sufficient number of pixels goes up with image size.
3. **Handling gigapixel WSI through divide and conquer:** As mentioned in Section 3.1, computation of  $W_s$  can easily be scaled up by limiting the number of pixels used for stain color basis estimation. On the other hand, steps that necessarily involve all pixels of the source image, e.g. computation of Beer-Lambert transform and inverse and the estimation of the normalized optical densities in equations 1, 3, and 4 were performed serially on patches of the whole WSI. This required time consuming disk reads and writes, which were marginally sped up by sampling longitudinal patches from a WSI instead of the usual square patches based on the insight about row-wise contiguous storage of a WSI on a hard drive.
4. **Efficient computation of order statistics:** As mentioned previously, some steps required computation of a 99<sup>th</sup>-percentile of stain density for outlier rejection, which we efficiently implemented for the WSI by taking medians of that percentile across patches.
5. **Speedup using TensorFlow:** First, re-implementation in python instead of Matlab itself led to a significant speedup for small images. Then, we implemented the heavy lifting computations of the algorithm in TensorFlow Abadi et al. [2016], which easily deploys any available CPU or GPU cores for speed.
6. **Software functionalities and easy debugging:** The new software is modular, easy to understand and user-friendly. The ability to normalize a batch of images using the same target image is also available for user efficiency, and so is a demo and a verbose mode for better understanding.

## 4 Results

We present an evaluation of the quality of the output normalized image of our improved SPCN algorithm compared to the original SPCN, and discuss the speed of our implementation on varying image sizes.

### 4.1 Quality of Color Normalization

A qualitative and quantitative comparison of the SPCN algorithm with other color normalization techniques is described in Vahadane et al. [2016], and justifies its choice as a starting point in this work. In Figure 2, we compare the performance of the original SPCN to our improved version on three challenging examples of source images taken from TCGA. On Image (a), the original SPCN exhibits swapping of stain color basis due to the presence of significant blue components in both stains. Our proposed improvement to compare the difference of red and blue components avoids this error. Image (b) illustrates that in cases where one stain dominates, SPCN can lead to a color tint in the intermediate

whitespace. Our algorithm avoids this by estimating a channel-wise maximum pixel intensity. Image (c) shows an image with a significant background portion, which leads to a strong tinge in the background after normalization in an extreme case, which is also handled well in our results by estimating the maximum intensity for each channel separately.

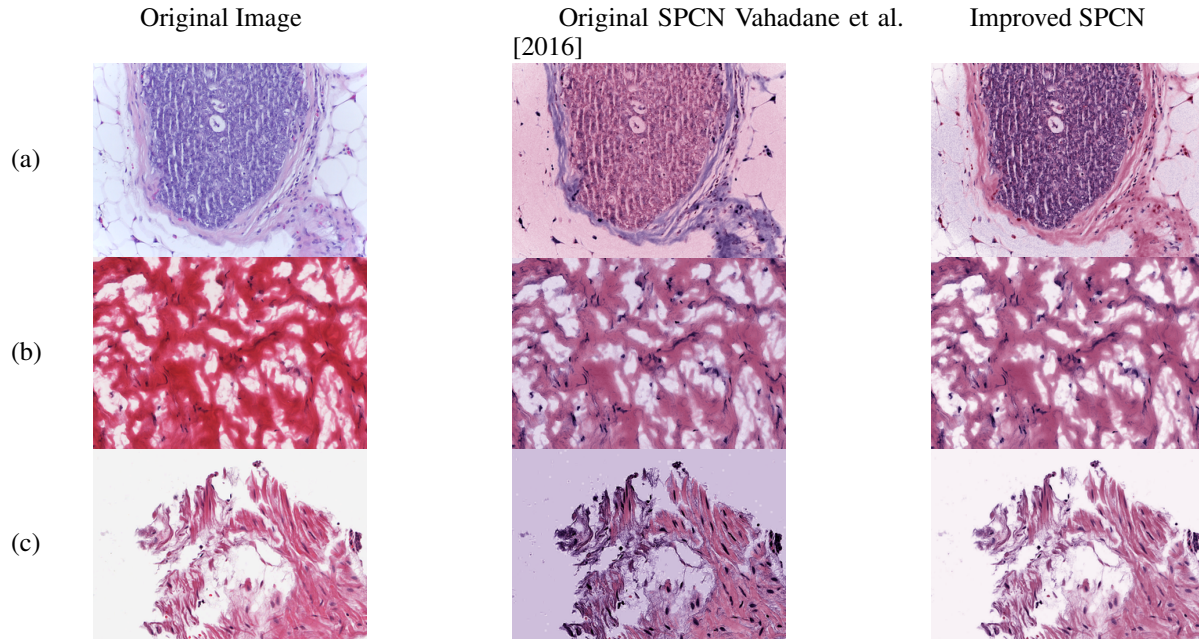


Figure 2: Comparison of image quality between original and improved SPCN.

## 4.2 Runtime analysis on gigapixel images

The flowchart in Figure 3 summarizes the steps of the algorithm, highlighting the efficiency achieved by our software through GPU computations. Even without GPU's, the TensorFlow implementation is much faster than the original Matlab implementation. A step-wise comparison of the time taken for color normalization by the almost original SPCN (modified to accommodate large images using python and NumPy) and our TensorFlow implementation using CPU and GPU respectively is shown in Figure 4. For relatively smaller images, the speeds of the implementations are comparable. Additionally, the time taken to estimate  $W_s$  remains almost constant for large image sizes due to our efficient implementation. As the images become very large, even though other steps scale linearly with image size, our algorithm achieves a multi-fold speedup over a simple python implementation. It takes only a few minutes for even 5 gigapixel images.<sup>3</sup> This shows that it is practical to use this software on large pathology images right out-of-the-box.

## 4.3 Comparison with other Deep learning methods

SPCN has been well validated against the most commonly used algorithms in the base paper Vahadane et al. [2016]. We compare the presented work to most recent literature Shaban et al. [2018], Zanjani et al. [2018] and Cho et al. [2017]. These papers present a Generative Adversarial Network (GAN) for color normalization. In Cho et al. [2017] the proposed method requires retraining once the target or reference image changes. In Zanjani et al. [2018], proposed method does not require the retraining when the reference image changes. In algorithm also presents many advantages over [BenTaieb] (which was published in March 2018). Firstly, the stain transfer procedure using GANs is a completely different approach to color normalization and it is a domain which is still largely unexplored and does not guarantee theoretical performance. Secondly, the algorithm does not perfectly preserve biological structure as it only approximates it using a loss function. Finally, it involves training of deep neural networks, requiring availability of training data, heavy computations, adjustment of hyperparameters and application to large images intractable. Moreover, it presents a black-box to the internal computations involved, making debugging difficult. Our algorithm in comparison, is well-validated, preserves structure, is simple to understand and debug, requires simple computations and with our software, can be used on images of any size. An additional advantage of our algorithm is that it contains a stain separation step as an integrated module, which can aid research work involving any particular stain in the histological images.

<sup>3</sup>Computer: CPU Core i7, 4.2GHz, 8 core, 64GB RAM; GPU Titan X 12GB RAM

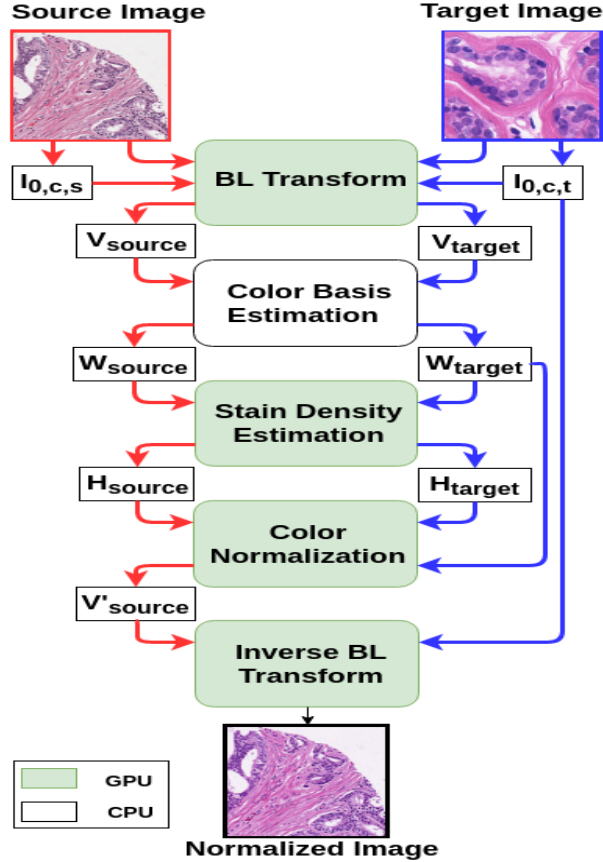


Figure 3: SPCN Flowchart

## 5 Conclusion

Uptake of computational pathology suffers from two major problems – lack of investment in whole slide scanners, and the lack of fast algorithms to process whole slide images. With this work, we have contributed to overcoming the second challenge by introducing an open-source software that can color normalize WSIs in a reasonable time. Our software significantly improved upon SPCN Vahadane et al. [2016] in this respect, while also improving on the image quality and retaining the advantages of SPCN. With the advent of deep learning, several research groups are using GPUs and open-source software such as OpenSlide Goode et al. [2013] and TensorFlow Abadi et al. [2016] to work with WSIs. Our improved and corrected implementation of SPCN is built on top of such libraries. To ensure that working with large slides does not seem daunting to medical researchers, it is necessary to make other widely-used parts of a computational pathology pipeline, such as nucleus detection and segmentation Kumar et al. [2017] and gland segmentation, also scalable and open-source.

## References

- Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.
- Hyungjoo Cho, Sunghin Lim, Gunho Choi, and Hyunseok Min. Neural stain-style transfer learning using gan for histopathological images. *arXiv preprint arXiv:1710.08543*, 2017.
- Adam Goode, Benjamin Gilbert, Jan Harkes, Drazen Jukic, and Mahadev Satyanarayanan. Openslide: A vendor-neutral software foundation for digital pathology. *Journal of pathology informatics*, 4, 2013.

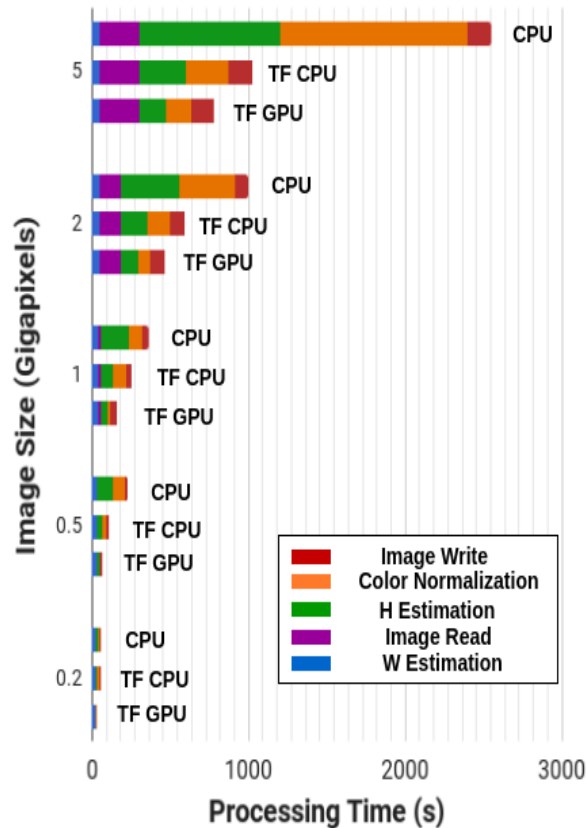


Figure 4: Comparison of time taken between python (CPU), TensorFlow without GPU (TF CPU), and with GPU (TF GPU)

Adnan Mujahid Khan, Nasir Rajpoot, Darren Treanor, and Derek Magee. A nonlinear mapping approach to stain normalization in digital histopathology images using image-specific color deconvolution. *IEEE Transactions on Biomedical Engineering*, 61(6):1729–1738, 2014.

Neeraj Kumar, Ruchika Verma, Sanuj Sharma, Surabhi Bhargava, Abhishek Vahadane, and Amit Sethi. A dataset and a technique for generalized nuclear segmentation for computational pathology. *IEEE transactions on medical imaging*, 36(7):1550–1560, 2017.

Marc Macenko, Marc Niethammer, James S Marron, David Borland, John T Woosley, Xiaojun Guan, Charles Schmitt, and Nancy E Thomas. A method for normalizing histology slides for quantitative analysis. In *Biomedical Imaging: From Nano to Macro, 2009. ISBI'09. IEEE International Symposium on*, pages 1107–1110. IEEE, 2009.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.

Amit Sethi, Lingdao Sha, Abhishek Ramnath Vahadane, Ryan J Deaton, Neeraj Kumar, Virgilia Macias, and Peter H Gann. Empirical comparison of color normalization methods for epithelial-stromal classification in h and e images. *Journal of pathology informatics*, 7, 2016.

M Tarek Shaban, Christoph Baur, Nassir Navab, and Shadi Albarqouni. Staingan: Stain style transfer for digital histological images. *arXiv preprint arXiv:1804.01601*, 2018.

Abhishek Vahadane, Tingying Peng, Amit Sethi, Shadi Albarqouni, Lichao Wang, Maximilian Baust, Katja Steiger, Anna Melissa Schlitter, Irene Esposito, and Nassir Navab. Structure-preserving color normalization and sparse stain separation for histological images. *IEEE transactions on medical imaging*, 35(8):1962–1971, 2016.

Farhad G Zanjani, Svitlana Zinger, Babak E Bejnordi, Jeroen AWM van der Laak, et al. Histopathology stain-color normalization using deep generative models. *MIDL*, 2018.